CISC 360, Fall 2008
Midterm Exam - SOLUTION
October 7, 2008

Instructions:

- Make sure that your exam is not missing any sheets, then write your full name on the front.
- Write your answers in the space provided below the problem. If you make a mess, clearly indicate your final answer.
- The exam has a maximum score of 100 points.
- The problems are of varying difficulty. The point value of each problem is indicated. Pile up the easy points quickly and then come back to the harder problems.
- This exam is OPEN BOOK. You can use any books. Good luck!

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(24):</td>
</tr>
<tr>
<td>2</td>
<td>(15):</td>
</tr>
<tr>
<td>3</td>
<td>(24):</td>
</tr>
<tr>
<td>4</td>
<td>(15):</td>
</tr>
<tr>
<td>5</td>
<td>(22):</td>
</tr>
<tr>
<td>TOTAL (100):</td>
<td></td>
</tr>
</tbody>
</table>
Problem 1. (24 points):
Consider the following datatype definitions on an IA32 (x86) machine under Windows.

```
typedef struct {
    char c;
    double *p;
    int i;
    double d;
    short s;
} struct1;
```

```
typedef union {
    char c;
    double *p;
    int i;
    double d;
    short s;
} union1;
```

A. Using the template below (allowing a maximum of 32 bytes), indicate the allocation of data for a structure of type `struct1`. Mark off and label the areas for each individual element (there are 5 of them). Cross hatch the parts that are allocated, but not used (to satisfy alignment).

Assume the alignment rules discussed in lecture: data types of size $x$ must be aligned on $x$-byte boundaries. Clearly indicate the right hand boundary of the data structure with a vertical line.

```
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
|---------------------------------------------------------------|
```

```
answer: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
|---------------------------------------------------------------|
```

B. How many bytes are allocated for an object of type `struct1`?

answer: 32

C. What alignment is required for an object of type `struct1`? (If an object must be aligned on an $x$-byte boundary, then your answer should be $x$.)

answer: 8

D. If we define the fields of `struct1` in a different order, we can reduce the number of bytes wasted by each variable of type `struct1`. What is the number of unused, allocated bytes in the best case?

answer: 5

E. How many bytes are allocated for an object of type `union1`?

answer: 8
F. What alignment is required for an object of type `union1`? (If an object must be aligned on an $x$-byte boundary, then your answer should be $x$.)

answer: 8
Problem 2. (15 points):
Consider the following 16-bit floating point representation based on the IEEE floating point format:

- There is a sign bit in the most significant bit.
- The next seven bits are the exponent. The exponent bias is 63.
- The last eight bits are the significand.

The rules are like those in the IEEE standard (normalized, denormalized, representation of 0, infinity, and NaN).
As described in Class 10, we consider the floating point format to encode numbers in a form:

\((-1)^s \times m \times 2^E\)

where \(m\) is the mantissa and \(E\) is the exponent.
Fill in the table below for the following numbers, with the following instructions for each column:

Hex: The 4 hexadecimal digits describing the encoded form.

\(m\): The fractional value of the mantissa. This should be a number of the form \(x\) or \(x/y\), where \(x\) is an integer, and \(y\) is an integral power of 2. Examples include: 0, 67/64, and 1/256.

\(E\): The integer value of the exponent.

Value: The numeric value represented. Use the notation \(x\) or \(x \times 2^z\), where \(x\) and \(z\) are integers.

As an example, to represent the number \(7/2\), we would have \(s = 0\), \(m = 7/4\), and \(E = 1\). Our number would therefore have an exponent field of \(0x40\) (decimal value \(63 + 1 = 64\)) and a significand field \(0xC0\) (binary 11000000), giving a hex representation \(40C0\).

You need not fill in entries marked “—”.

<table>
<thead>
<tr>
<th>Description</th>
<th>Hex</th>
<th>(m)</th>
<th>(E)</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>(-0)</td>
<td>8000</td>
<td>0</td>
<td>-62</td>
<td>-0</td>
</tr>
<tr>
<td>Smallest value &gt; 1</td>
<td>3F01</td>
<td>257/256</td>
<td>0</td>
<td>257/256</td>
</tr>
<tr>
<td>Largest Denormalized</td>
<td>00FF</td>
<td>255/256</td>
<td>-62</td>
<td>255 \times 2^{-70}</td>
</tr>
<tr>
<td>(-\infty)</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Number with hex representation 3AA0</td>
<td>—</td>
<td>13/8</td>
<td>-5</td>
<td>13/256</td>
</tr>
</tbody>
</table>
Problem 3. (24 points):
Consider a 6-bit two’s complement representation. Fill in the empty boxes in the following table:

<table>
<thead>
<tr>
<th>Number</th>
<th>Decimal Representation</th>
<th>Binary Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zero</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>n/a</td>
<td>-1</td>
<td></td>
</tr>
<tr>
<td>n/a</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>n/a</td>
<td>-10</td>
<td></td>
</tr>
<tr>
<td>n/a</td>
<td>01 1010</td>
<td></td>
</tr>
<tr>
<td>n/a</td>
<td>10 0110</td>
<td></td>
</tr>
<tr>
<td>Tmax</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tmin</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tmax+TMax</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tmin+TMin</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tmin+1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tmin−1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tmax+1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>−TMax</td>
<td></td>
<td></td>
</tr>
<tr>
<td>−TMin</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**Answer:**

<table>
<thead>
<tr>
<th>number</th>
<th>decimal</th>
<th>binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zero</td>
<td>0</td>
<td>00 0000</td>
</tr>
<tr>
<td>n/a</td>
<td>-1</td>
<td>11 1111</td>
</tr>
<tr>
<td>n/a</td>
<td>5</td>
<td>00 0101</td>
</tr>
<tr>
<td>n/a</td>
<td>-10</td>
<td>11 0110</td>
</tr>
<tr>
<td>n/a</td>
<td>26</td>
<td>01 1010</td>
</tr>
<tr>
<td>n/a</td>
<td>-26</td>
<td>10 0110</td>
</tr>
<tr>
<td>Tmax</td>
<td>31</td>
<td>01 1111</td>
</tr>
<tr>
<td>Tmin</td>
<td>-32</td>
<td>10 0000</td>
</tr>
<tr>
<td>Tmax+TMax</td>
<td>-2</td>
<td>11 1110</td>
</tr>
<tr>
<td>Tmin+TMin</td>
<td>0</td>
<td>00 0000</td>
</tr>
<tr>
<td>Tmin+1</td>
<td>-31</td>
<td>10 0001</td>
</tr>
<tr>
<td>Tmin-1</td>
<td>31 (Tmax)</td>
<td>01 1111</td>
</tr>
<tr>
<td>Tmax+1</td>
<td>-32 (Tmin)</td>
<td>10 0000</td>
</tr>
<tr>
<td>-TMax</td>
<td>-31</td>
<td>10 0001</td>
</tr>
<tr>
<td>-TMin</td>
<td>-32</td>
<td>10 0000</td>
</tr>
</tbody>
</table>
Problem 4. (15 points):
Consider the following assembly representation of a function foo containing a for loop:

```
foo:
    pushl %ebp
    movl %esp,%ebp
    pushl %ebx
    movl 8(%ebp),%ebx
    leal 2(%ebx),%edx
    xorl %ecx,%ecx
    cmpl %ebx,%ecx
    jge .L4
    .L6:
        leal 5(%ecx,%edx),%edx
        leal 3(%ecx),%eax
        imull %eax,%edx
        incl %ecx
        cmpl %ebx,%ecx
        jl .L6
    .L4:
        movl %edx,%eax
        popl %ebx
        movl %ebp,%esp
        popl %ebp
    ret
``` 

Fill in the blanks to provide the functionality of the loop:

```c
int foo(int a)
{
    int i;
    int result = _____________;

    for( ________; ________; i++ ) {
        _____________________;
        _____________________;
    }
    return result;
}
```

```c
int foo(int a)
{
    int i;
    int result = a + 2;

    for (i=0; i < a; i++) {
        result += (i + 5);
        result *= (i + 3);
    }
    return result;
}
```
This next problem will test your understanding of stack frames. It is based on the following recursive C function:

```c
int silly(int n, int *p)
{
    int val, val2;

    if (n > 0)
        val2 = silly(n << 1, &val);
    else
        val = val2 = 0;

    *p = val + val2 + n;

    return val + val2;
}
```

This yields the following machine code:

```assembly
silly:
    pushl %ebp
    movl %esp,%ebp
    subl $20,%esp
    pushl %ebx
    movl 8(%ebp),%ebx
    testl %ebx,%ebx
    jle .L3
    addl $-8,%esp
    leal -4(%ebp),%eax
    pushl %eax
    leal (%ebx,%ebx),%eax
    pushl %eax
    call silly
    jmp .L4
.p2align 4,,7
.L3:
    xorl %eax,%eax
    movl %eax,-4(%ebp)
.L4:
    movl -4(%ebp),%edx
    addl %eax,%edx
    movl 12(%ebp),%eax
    addl %edx,%ebx
    movl %ebx,(%eax)
    movl -$24(%ebp),%ebx
    movl %edx,%eax
    movl %ebp,%esp
    popl %ebp
    ret
```
Problem 5. (22 points):

A. Is the variable `val` stored on the stack? If so, at what byte offset (relative to `%ebp`) is it stored, and why is it necessary to store it on the stack? answer: yes, -4, Need to pass pointer to it to recursive call.

B. Is the variable `val2` stored on the stack? If so, at what byte offset (relative to `%ebp`) is it stored, and why is it necessary to store it on the stack? answer: no.

C. What (if anything) is stored at `-24 (%ebp)`? If something is stored there, why is it necessary to store it? answer: the value of ebx is saved here, because ebx is a callee-save register.

D. What (if anything) is stored at `-8 (%ebp)`? If something is stored there, why is it necessary to store it? answer: nothing is stored here.